Symbiot: Congestion-driven Multi-resource Fairness for Multi-User Sensor Networks

Yad Tahir*, Shusen Yang†, Usman Adeel*, Julie McCann*
*Imperial College London, †University of Liverpool
{yst11, u.adeel09, j.mccann}@imperial.ac.uk, shusen.yang@liverpool.ac.uk

***Abstract*—In this paper, we study the problem of *multi-resource* fairness in multi-user sensor networks with heterogeneous and time-varying resources. Particularly we focus on data gathering applications run on Wireless Sensor Networks (WSNs) or Internet of Things (IoT) in which users require to run a serious of sensing operations with various resource requirements. We consider both the resource demands of *sensing tasks*, and *data forwarding tasks* needed to establish multi-hop relay communications. By exploiting graph theory, queueing theory and the notion of *dominant resource shares*, we develop *Symbiot*, a light-weight, distributed algorithm that ensures multi-resource fairness between these users. With Symbiot, nodes can independently schedule its resources while maintaining network-level resource fairness through observing *traffic congestion levels*. Large-scale simulations based Contiki OS and Cooja network emulator show the effectiveness of Symbiot in adaptively utilizing available resources and reducing average completion times.**

I. Introduction

Wireless Sensor Networks (WSNs) and Internet of Things (IoT) [1] are evolving towards interconnected, sensing and processing infrastructures that are expected to provide services for multiple concurrent users. One of the main reasons of having such shared enterprise deployments is to maximize the return on investment while minimizing operating costs.

Such multi-user networks are almost invariably heterogeneous in terms of their hardware components, offered resource capacities as well as user demands. Various types of sensors can be attached to nodes. The assorted capacities of some resource types, such as bandwidth, can be highly dynamic and time-varying [2], [3]. In addition, diversity is found when users demand specific resources. For instance, a user may require the execution of a memory-heavy task on nodes with temperature sensors, while another user may need to compute a bandwidth-hungry task on nodes having humidity and light sensors.

Resource sharing in any multi-user sensor network is a key issue for one primary reason. The rapid growth of traffic and computation requirements are often not matching the growth and expansion of overall network capacity. Hence, the limited network resources should be distributed *fairly* between users [4], [5]. Accounting for diversity across node resources and user requirements presents an increased challenge to schedulers for fair provisioning of resources. Getting this right is fundamental to next-generation WSN and IoT systems.

This paper concerns how the available network resources, including those that are time-varying, should be allocated between competing users. In particular, we consider the problem of fair resource allocation for data gathering applications in WSNs and IoT.

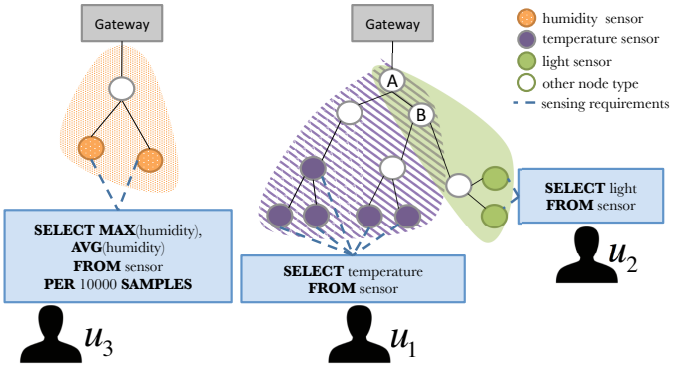

Fig. 1. A WSN with two gateways and three users. u_1 and u_2 are interested in nodes with temperature and light sensors, respectively. u_3 needs to execute a query with higher computation complexity on nodes with humidity sensors.

A. Motivation

Both resource and demand heterogeneity have an impact on the efficient and fair allocation of resources to users. Current *max-min*-based resource allocation schemes such as [6] do not deal well with both heterogeneous resources and user demands in multi-resource systems. Utilizing *multi-resource fairness* schemes [4], [5] such as *Dominant Resource Fairness (DRF)* [4] have become a hot topic in both computation economics communities and cloud computing. Given users with heterogeneous demands, these approaches achieve more efficient allocation than *single-resource fairness* schemes.

However, these approaches are limited when it comes to providing *distributed* scheduling that can adapt to time-varying resources. They assume that the system has a single, *centralized* scheduler that is aware about the resource capacities of all the system components. This means in a network with dynamic resources, nodes have to repeatedly report their available resource capacities to the scheduler. This is neither scalable nor resilient, thus impractical for large-scale networks. Furthermore, these approaches do not address multi-hop nature on which the network relies. When a user executes a task on a particular node, the output has to be forwarded to a *gateway device* using multi-hop communications. This creates *indirect* resource consumptions from the user. To ensure having fair multi-resource scheduling, it is crucial to consider both *direct* and *indirect* resource demands in multi-hop networks.

To understand the complexity of acheiving multi-resource fairness in multi-user sensor networks, consider a typical sensing application shown in Fig. 1. The network has three users u_1, u_2, and u_3 with different *sensing requirements* expressed as SQL-like queries for convenience. Two cases need to be analyzed here: first, based on the sensing requirements, u_1 and u_2 do not compete on resources. Each of them requires a different set of nodes. However the employed multi-hop communications make u_1 and u_2 *indirectly* related to each other. They now compete for the shared resources of rely nodes (nodes A and B). The second case is that if u_1 changed its requirements and asked for all the resources available in all nodes with temperature or light sensors, surely this would be considered *unfair* according to the max-min fairness model as u_1 would consume considerably more resources than u_2.

From these two cases there are two key questions that have to be carefully addressed: first, *how can we formally define the relationship between users?* u_1 and u_2 are dependent in terms of resource requirements, but u_3 certainly is not. Second, given each user submit sensing tasks to the network, *what is the maximum number of tasks that each user can execute while the resource consumptions between related users remain fair?*

B. Contributions

We present the study of the fair resource allocation problem in network with multiple, time-varying resource types. We summarize the contributions of this paper as follows:

1. Section II establishes a set of system models to formally describe multi-user WSNs and IoT. By exploiting graph theory, we rigorously define the relationship between users in terms of their *direct* and *indirect* resource requirements.

2. Section III proposes a new extension to max-min fairness that exploits graph theory and dominant resource sharing to ensure multi-resource fairness between users running a series of sensing tasks. We develop a new algorithm, named Symbiot, that uses iterative linear programming and queueing theory to compute the proposed extension. Symbiot is not only lightweight, but also it is fully distributed eliminating the need for centralized scheduling. Through observing local resource capacities and current *traffic congestion levels*, nodes can independently allocate resources for its users while maintaining multi-resource fairness between related users in the whole network. To the best of our knowledge, this work is not only the first work to address the *indirect* requirement problem and utilize queueing theory in multi-resource scheduling, but also there has been no prior work that has proposed a distributed solution to achieve multi-resource fairness.

3. In Section IV, we implemented Symbiot on Contiki OS [7], an open source operating system for wireless sensor networks and IoT. We integrated Symbiot with the IPv6 stack provided by Contiki. Through large-scale simulations with the Cooja network emulator [7], we demonstrate the practical performance of Symbiot algorithm. Our experiments show that Symbiot outperforms the naive version of distributed DRF in terms of reducing the average of job completion times.

II. System Models

We consider a WSN or IoT that consists of a set of wireless nodes $\mathcal{N} = \mathcal{S} \cup \mathcal{H}$ communicating in a multi-hop fashion, where $\mathcal{S}$ represents the set of all nodes that can sense, generate and relay data packets; while $\mathcal{H}$ is the set of all gateways that collect the data traffic produced by the network. Let the set $\mathcal{N}_x$ holds all one-hop neighbors that node $x \in \mathcal{N}$ that can communicate with. The network has a set of users $\mathcal{U}$ and we assume that the network operates in discrete time with a unit time slot (e.g. a second) $t \in \{1, 2, ...\}$.

A. Node Operations and Data Queues

Node $n \in \mathcal{S}$ can execute two types of tasks: *sensing* and *data forwarding*. Each user $u \in \mathcal{U}$ can run *sensing tasks* periodically on nodes. We assume that a sensing task requires no more than one time slot to finish. We also assume that the sensing tasks are delay-tolerant, which is realistic as many monitoring applications are reasonably delay tolerant.

Performing sensing tasks will generate data packets to be collected by a gateway $h \in \mathcal{H}$. To achieve this, nodes execute *data forwarding tasks*. The data can be transmitted directly if $h \in \mathcal{N}_n$, or indirectly in a multi-hop fashion if $h \notin \mathcal{N}_n$. The underlying routing layer determines the forwarding policy. In this paper, we only consider networks running in a fixed routing scenario, i.e., the route of each data flow is predetermined and fixed during the time of interest. We also assume both sensing and forwarding tasks are *indivisible*.

In the network, each node maintains data packets through queues (data buffers). Let Q_n^u denote the queue existing in node n to hold the data packets generated by user u. Let $Q_n^u(t)$ be the queue backlog (or queue length) of the queue Q_n^u at time slot t. The queue dynamics of Q_n^u are defined as follows:

$$\begin{aligned} &0 \leq Q_n^u(t) \leq MaxQ_n^u \\ &Q_n^u(t+1) = |Q_n^u(t) - f_{u,n}^{out}(t)|_+ + r_{u,n}(t) + f_{u,n}^{in}(t) \end{aligned} \quad (1)$$

Where the operator $|\ .\ |_+$ represents $max(0, .)$ and $MaxQ_n^u > 0$ is the maximum queue length (i.e. allocated data buffer size) of Q_n^u. $r_{u,n}(t)$ is the output of performing sensing tasks in node n for user u at time slot t. $f_{u,n}^{in}(t)$ and $f_{u,n}^{out}(t)$ are the incoming and outgoing traffic of node n for user u, respectively. It is worth noting that the queue length of any gateway always equals to zero.

B. Node Resources

Each node $n \in \mathcal{N}$ offers K types of resources for users such as computing (i.e. MCU), memory, bandwidth, and light sensors. Let a K-dimensional vector $\mathbf{c}_n(t) = (c_{n,1}(t), ..., c_{n,K}(t))$ represents node's resource capacities at time slot t, where each entry $c_{n,k}(t)\ 1 \leq k \leq K$ represents the resource capacity of kth resource of n at t. Let $\mathbf{C}(t)$ be a $|\mathcal{N}| \times K$-dimensional matrix representing the resource capacity for the whole network at t.

In the rest of this paper, we will use a tuple $(n, k),\ n \in \mathcal{N}, 1 \leq k \leq K$ to refer to a specific resource.

C. User Resource Requirements

Let $\widehat{\mathbf{R}}^u$ be a $|\mathcal{N}|\times\ K$-dimensional matrix in which the entry $\widehat{\mathbf{R}}^u_{n,k}$ represents the requirement from (n,k) to perform a sensing task by user u. Let $\widetilde{\mathbf{R}}^u$ be a $|\mathcal{N}|\times\ K$-dimensional matrix in which the entry $\widetilde{\mathbf{R}}^u_{n,k}$ is the amount of resource needs from (n,k) to perform a data forwarding task on Q^u_n.

The *gross resource requirements* of user u is represented as a $|\mathcal{N}|\times\ K$ matrix $\mathbf{R}^u$, where:

$$\mathbf{R}^u_{n,k} = \widehat{\mathbf{R}}^u_{n,k} + \widetilde{\mathbf{R}}^u_{n,k} \tag{2}$$

Consider $\mathcal{D}_u$ to represent the set of *strictly positive resource demands* of user u:

$$\mathcal{D}_u := \{(n,k) : \mathbf{R}^u_{n,k} > 0,\ n \in \mathcal{N},\ 1 \le k \le K\} \tag{3}$$

D. Topological User Dependency

This section presents a graph model to characterize the dependency of all users in $\mathcal{U}$, with respect to resource requirements.

Definition 1. [User Dependency Graph]. We can use an undirected graph $G(\mathcal{U},\mathcal{V})$ to represent the dependency of all users, where

$$\mathcal{V} := \{(u,v) : u,v \in \mathcal{U},\ \mathcal{D}_u \cap \mathcal{D}_v \neq \emptyset\}$$

is the set of the links in the graph. Each link $(u,v) \in \mathcal{V}$ indicates that two users u and v share some resources.

Definition 2. [User Dependent Cluster (UDC)]. For a given user dependence graph $G(\mathcal{U},\mathcal{V})$, a User Dependent Cluster (UDC) $\mathcal{C} \subseteq \mathcal{U}$ is defined as the set of users in a connected component[1] of $G(\mathcal{U},\mathcal{V})$.

Define $\mathcal{U}_{n,k}$ as the set of users that have a *strictly positive demand* of resource $(n,k),\ n \in \mathcal{N}, 1 \le k \le K$, i.e.

$$\mathcal{U}_{n,k} := \{u : \mathbf{R}^u_{n,k} > 0,\ u \in \mathcal{U}\} \tag{4}$$

It can be seen that two users u_1 and u_2 in a UDC $\mathcal{C}$ may not share common resources, meaning that they are not *directly dependent* on each other. However the resource requirements of u_1 would affect that of all neighbors $v \in \mathcal{U}_{n,k},\ (n,k) \in \mathcal{D}_{u_1}$. By repeating the above process, the requirement of u_1 would affect all users in $\mathcal{C}$, including u_2. This implies that all users in a UDC are *directly or indirectly dependent* in terms of resource requirements.

E. Objective

The objective of this paper is to develop a distributed, fair scheduling algorithm to allocate resources for sensing and data forwarding tasks for the users of the network. In particular for every time slot t, it is required to distributedly compute two $|\mathcal{N}| \times |\mathcal{U}|$–dimensional matrices: $\widehat{\mathbf{X}}(t)$ and $\widetilde{\mathbf{X}}(t)$, where the entries $\widehat{\mathbf{X}}_{u,n}(t)$ and $\widetilde{\mathbf{X}}_{u,n}(t)$ are the number of sensing and forwarding tasks performed by user u in node n, respectively.

[1]In graph theory, a connected component is subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the supergraph.

The network resources allocated to user u at time slot t is defined as a $|\mathcal{N}|\times\ K$–dimensional matrix $\mathbf{A}^u(\widehat{\mathbf{X}}(t),\widetilde{\mathbf{X}}(t))$ where each entry $\mathbf{A}^u_{n,k}(t)$ represents the amount of resource (n,k) allocated to user u at time slot t:

$$\mathbf{A}^u_{n,k}(t) = \widehat{\mathbf{X}}_{u,n}(t)\widehat{\mathbf{R}}^u_{n,k} + \widetilde{\mathbf{X}}_{u,n}(t)\widetilde{\mathbf{R}}^u_{n,k} \tag{5}$$

Definition 3. [Feasible Resource Allocation]. *A resource allocation matrix $\boldsymbol{A}^u(\widehat{\mathbf{X}}(t),\widetilde{\mathbf{X}}(t))$ is feasible if the following condition is satisfied:*

$$\sum_{u\in\mathcal{U}} \mathbf{A}^u(\widehat{\mathbf{X}}(t),\widetilde{\mathbf{X}}(t)) \preceq \mathbf{C}(t) \tag{6}$$

where $\preceq$ means entry-wise smaller than or equal to.

III. SYMBIOT

Let $d_{u,n,k}(t) = \mathbf{R}^u_{n,k}/c_{n,k}(t)$ be the *resource demand share* of user u for resource (n,k) at slot t. Let $d^{\max}_{u,n}(t)$ denote the *node-wise dominant demand share* of user u in node n.

$$d^{\max}_{u,n}(t) = \max_{1\le k\le K} d_{u,n,k}(t) \tag{7}$$

In Symbiot we *normalize* the resource demand shares of users in respect to their node-wise dominant demand share as follow:

$$d_{u,k}(t) = d_{u,n,k}(t)/d^{\max}_{u,n}(t) \tag{8}$$

For each user u in a UDC $\mathcal{C}$, define *UDC-wise dominant demand share* as

$$d^{\max}_u(t) = \max_{(n,k)\in\mathcal{D}(\mathcal{C})} d^{\max}_{u,n}(t),\ u \in \mathcal{C} \tag{9}$$

where

$$\mathcal{D}(\mathcal{C}) := \bigcup_{u\in\mathcal{C}} \mathcal{D}_u$$

denotes the set of all resources used by all users in $\mathcal{C}$. For a given $d^{\max}_u(t)$, we denote the node $n^* = \arg\max_n d^{\max}_{u,n}(t)$ as the *bottleneck node* for user u in $\mathcal{C}$. We define the *UDC-wise dominant share* for user u in $\mathcal{C}$ as:

$$d^{\max}_u(t)(\widehat{\mathbf{X}}_{u,n^*}(t) + \widetilde{\mathbf{X}}_{u,n^*}(t)) \tag{10}$$

A. Algorithm

The basic idea of Symbiot is to implement a distributed, light-weight algorithm to approximate the max-min fairness for the UDC-wise dominant shares for each UDC.

Because the algorithm is distributed, every node $n \in \mathcal{N}$ computes resource scheduling locally. However, we utilize network traffic congestion to collect feedback from the bottleneck nodes existing in UDCs. The pseudo code of Symbiot is summarized in Fig. 2 and Fig. 3.

In node n at the beginning of time slot t, all users that use n are considered to be unsaturated (line 01 in Fig. 2) and stored in the unsaturated user set $\acute{\mathcal{U}}_{us}$. Symbiot allocates resources for the unsaturated users through multiple iterations

Variables:
t: current time slot for the node
n: the current node
$\acute{\mathcal{U}}_{us}$: the set of unsaturated users.
$\acute{\mathcal{U}}_{pt}$: the set of users that performed a task in an iteration.
$\mathbf{B}$: a $|\mathcal{U}|\times\ K$ matrix holding resource budgets for an iteration
$\acute{\mathbf{A}}$: a $|\mathcal{U}|\times\ K$ matrix holding allocated resources for an iteration.

Input:
$\widehat{\mathbf{R}}_n := \{\widehat{\mathbf{R}}^u_{n,k},\ \ \forall u \in \mathcal{U}, 1 \leq k \leq K\}$
$\widetilde{\mathbf{R}}_n := \{\widetilde{\mathbf{R}}^u_{n,k},\ \ \forall u \in \mathcal{U}, 1 \leq k \leq K\}$

Output:
$\widehat{\mathbf{X}}_{u,n}(t), \widetilde{\mathbf{X}}_{u,n}(t)\ \ \forall u \in \mathcal{U}$

Main Algorithm:
01: $\acute{\mathcal{U}}_{us} \leftarrow \{u : \mathcal{U}_{n,k},\ \forall k\}$; // $\mathcal{U}_{n,k}$ defined in Eq. 4
02: **while** $\acute{\mathcal{U}}_{us}(t) \neq \emptyset$ **do**
03: $\acute{\mathbf{A}} \leftarrow 0, \mathcal{U}_{pt} \leftarrow \emptyset$; //reset the paramaters for this iteration
04: **compute** $d_{u,k}(t), \forall k, u \in \mathcal{U}_{us}$; //based on Eq. 8
05: $b \leftarrow 1/\max_{\forall k} \sum_{u \in \acute{\mathcal{U}}_{us}} d_{u,k}(t)$;
//node-wise dominant demand share maximization
06: **for all** $u \in \acute{\mathcal{U}}_{us}(t)$
07: $\mathbf{B}_{u,k} \leftarrow d_{u,n}(t) c_{n,k}(t) b, \forall k$; //assign resource budgets
08: **if** ***containLocalPacket***$(Q^u_n(t)) =$ **FALSE**
09: $[\ r\ \acute{\mathbf{A}}\ \mathcal{U}_{pt}\ ] \leftarrow$ ***sensing***$(u, \acute{\mathbf{A}}, \mathcal{U}_{pt}, \widehat{\mathbf{R}}_n, \mathbf{B})$;
10: $\widehat{\mathbf{X}}_{u,n}(t) \leftarrow \widehat{\mathbf{X}}_{u,n}(t) + r$;
12: **end if**
13: $r \leftarrow 1$;
14: **while** $r = 1$
15: $[\ r\ \acute{\mathbf{A}}\ \mathcal{U}_{pt}\ ] \leftarrow$ ***dataForwarding***$(u, \acute{\mathbf{A}}, \mathcal{U}_{pt}, \widetilde{\mathbf{R}}_n, \mathbf{B})$;
16: $\widetilde{\mathbf{X}}_{u,n}(t) \leftarrow \widetilde{\mathbf{X}}_{u,n}(t) + r$;
17: **end while**
18: **end for**
19: $c_{n,k}(t) \leftarrow c_{n,k}(t) - \sum_{u \in \acute{\mathcal{U}}_{us}} \acute{\mathbf{A}}_{u,k}, \forall k$;
//update available resources based on actual consumptions
20: $\acute{\mathcal{U}}_{us} \leftarrow \acute{\mathcal{U}}_{us} \cap \acute{\mathcal{U}}_{pt}$; // update unsaturated users
21: **end while**

Fig. 2. Symbiot - Main Algorithm

(lines 02-21 in Fig. 2). In each iteration Symbiot finds a budget b that maximizes the equalized node-wise dominant demand shares of these users (lines 04-05 in Fig. 2) This computation also ensures that Symbiot adheres to the capacity constraint introduced in the *Definition 3*.

Then Symbiot assigns each unsaturated user a resource budget for every resource type k (line 07 in Fig. 2). The given resource budgets can be utilized for sensing tasks, data forwarding tasks or both. For sensing tasks, Symbiot first checks whether the data queue of the user contains a local packet (i.e. generated by n). If no local packet found, Symbiot executes the $sensing$ function. It is worth noting that in the pseudo code of Symbiot, we assume the data queues of the users follow the First-In-First-Out (FIFO) scheduling. Symbiot utilizes the remaining budget for data forwarding tasks by calling the $dataForwarding$ function. As shown in Fig. 3, both $sensing$ and $dataForwarding$ update the resource

sensing$(u, \acute{\mathbf{A}}, \mathcal{U}_{pt}, \widehat{\mathbf{R}}_n, \mathbf{B})$
01: **if** $Q^u_n(t) \neq MaxQ^u_n \wedge \exists k\ \widehat{\mathbf{R}}^u_{n,k} \neq 0 \wedge$
$\forall k\ \acute{\mathbf{A}}_{u,k} + \widehat{\mathbf{R}}^u_{n,k} \leq \mathbf{B}_{u,k}$
//if Q^u_n is not full, user requires sensing, and sufficient budget
02: $\acute{\mathbf{A}}_{u,k} \leftarrow \acute{\mathbf{A}}_{u,k} + \widehat{\mathbf{R}}^u_{n,k}, \forall k$; //allocate the resources
03: ***executeSensingTask***(u); //sensing and storing data in Q^u_n
04: $\acute{\mathcal{U}}_{pt}(t) \leftarrow \acute{\mathcal{U}}_{pt}(t) \cup \{u\}$;
05: **return** $[\ 1\ \acute{\mathbf{A}}\ \mathcal{U}_{pt}\ ]$;
06: **end if**
07: **return** $[\ 0\ \acute{\mathbf{A}}\ \mathcal{U}_{pt}\ ]$;

dataForwarding$(u, \acute{\mathbf{A}}, \mathcal{U}_{pt}, \widetilde{\mathbf{R}}_n, \mathbf{B})$
08: $p \leftarrow$ ***getNextHop***(); //node to where packets should be sent
09: **if** $Q^u_n(t) \neq 0 \wedge Q^u_p(t) \neq MaxQ^u_p \wedge \forall k\ \acute{\mathbf{A}}_{u,k} + \widetilde{\mathbf{R}}^u_{n,k} \leq \mathbf{B}_{u,k}$
//check Q^u_n is not empty, Q^p_n is not full, and sufficient budget
10: $\acute{\mathbf{A}}_{u,k} \leftarrow \acute{\mathbf{A}}_{u,k} + \widetilde{\mathbf{R}}^u_{n,k}, \forall k$; //allocate the resources
11: ***sendDataPacket***(u, p); //sends a packet from Q^u_n to p
12: $\acute{\mathcal{U}}_{pt} \leftarrow \acute{\mathcal{U}}_{pt} \cup \{u\}$;
13: **return** $[\ 1\ \acute{\mathbf{A}}\ \mathcal{U}_{pt}\ ]$;
14: **end if**
15: **return** $[\ 0\ \acute{\mathbf{A}}\ \mathcal{U}_{pt}\ ]$;

Fig. 3. Symbiot - Sensing and Data Forwarding functions

consumptions and active users set $\acute{\mathcal{U}}_{pt}$ for the current iteration. These functions returns 1 if the sensing or data forwarding tasks is actually executed, otherwise 0. Based on the feedback given in the lines 09-10 and 15-16 in Fig. 2, Symbiot updates the $\widehat{\mathbf{X}}_{u,n}(t)$ and $\widetilde{\mathbf{X}}_{u,n}(t)$ outputs for all unsaturated users. Then the remaining resource capacities are adjusted according to the resource consumptions of this iteration. In line 20 in Fig. 2, Symbiot removes *saturated users* from $\acute{\mathcal{U}}_{us}$. We define a saturated user as a user who did not perform any task, whether sensing or data forwarding, in an iteration. This process continues until all users become saturated.

Remark 1. *Both sensing and data forwarding operations are traffic-aware*. Even if a user has the required budget to perform a task, Symbiot will not allocate resources if traffic congestion is observed for that user. In line 01 in Fig. 3, sensing tasks check whether the queue of the current node is not full. Line 09 in Fig. 3 ensures data forwarding tasks are not performed if the queue of the next-hop node is not full.

Remark 2. *In a bottleneck node, the data queues of some users will be full*. A bottleneck node n^* in a UDC will limit the budget of one or more users as they are seen to be locally unfair with respect to other users. This means in the node n^*, the $f^{out}_{u,n^*} < f^{in}_{u,n^*}$ for some users. Based on the queue dynamics defined in Eq. 1, the queue levels of some $Q^u_{n^*}$ will gradually increase until it reaches the corresponding $MaxQ^u_{n^*}$.

Remark 3. *Bottleneck nodes in a UDC cause traffic congestion for nodes sending data packets in*. According to *Remark 2*, when $Q^u_{n^*} = MaxQ^u_{n^*}$, n^* also limits data forwarding operations for nodes that send data directly to n^* as described in *Remark 1*. Eventually when their data queues become full, this results in affecting the sensing operations performed on these nodes. This process repeats itself throughout the UDC and will affect all nodes that generating and sending data packet for user u through the n^*.

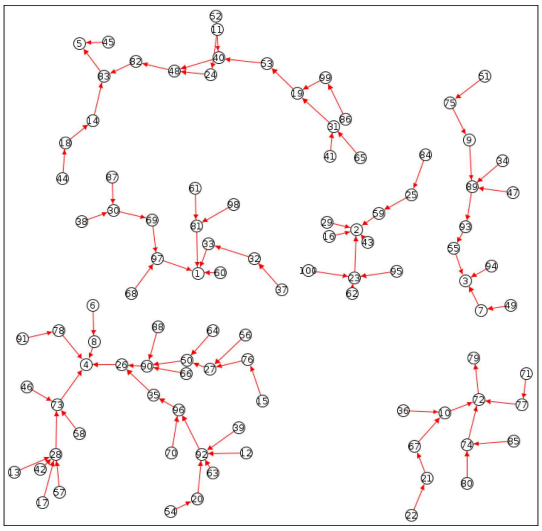

Fig. 4. The network topology of the experiments.

IV. Evaluation

The performance of Symbiot was evaluated through conducting a set of simulation on Cooja. We implemented Symbiot on top of Contiki OS [7] and its IPv6 stack. The implementation provides a set of public APIs to allow users to express their sensing and data forwarding requirements. Internally it uses UDP messages and IPv6 to transfer data packets from nodes to gateways. We use ICMPv6 to periodically broadcast the queue lengths of the users between nodes.

We established a 100-node network with 5 gateways and 95 nodes in a 600m×600m area. The network topology of our experiments is shown in Fig. 4. The network uses CSMA and RPL with the default parameter settings for the MAC and routing layers, respectively. We set RPL to use the ETX *Objective Function*. The nodes employ IEEE 802.15.4 transceiver, CC2420 with transmission range of 50m. Based on the experimental studies in [8], the bandwidth capacity of the nodes is set to be 160 40-bytes packets per second. The nodes in our network uses MSP430F1611 microcontroller with 10KB RAM. The network offers two types of sensors: temperature and humidity with the resource capacities of 1000 and 2000 readings per second, respectively. We randomly chose 40 nodes to offer temperature sensing and another 40 nodes for humidity sensing. In all simulations the duration of a time slot was set as one second and $MaxQ_n^u$ for all users in all nodes were equal to 70.

In the reset of the paper, we will use the vector $\langle r_1$ packets, r_2 bytes of RAM, r_3 temperature readings, r_4 humidity readings$\rangle$ to represent the resource requirements of offered resource types in the experiments.

A. Dynamic Resource Allocation

In our first experiment, we show how Symbiot dynamically allocates resources between users in a system with time-varying resource requirements and capacities. Fig. 5 shows the average allocated network resources and UDC-wise shares for each users as a function of time.

In the beginning, the network has two users u_1 and u_2. u_1 has the resource requirements of $\langle 0, 1, 2, 0\rangle$ for sensing and $\langle 1, 20, 0, 0\rangle$ for data forwarding. Similarly, u_2 needs $\langle 0, 10, 0, 20\rangle$ for sensing and $\langle 2, 50, 0, 0\rangle$ for data forwarding operations. As seen in Fig. 5, the UDC-wise dominant shares for both users are equal to 50% as bandwidth is the most demanded resource for both users in bottleneck nodes. At slot 300, u_2 changes his/her resource requirements making it more memory demanding. u_2 now demands $\langle 0, 2048, 200, 0\rangle$ for sensing and $\langle 9, 512, 0, 0\rangle$ for data forwarding. Symbiot dynamically adjusts the resource allocation for both users. The UDC-wise dominant shares increased to around 60% as the most required resource for u_1 is bandwidth while for u_2 now it becomes RAM. At time slot 600, u_3 enters the

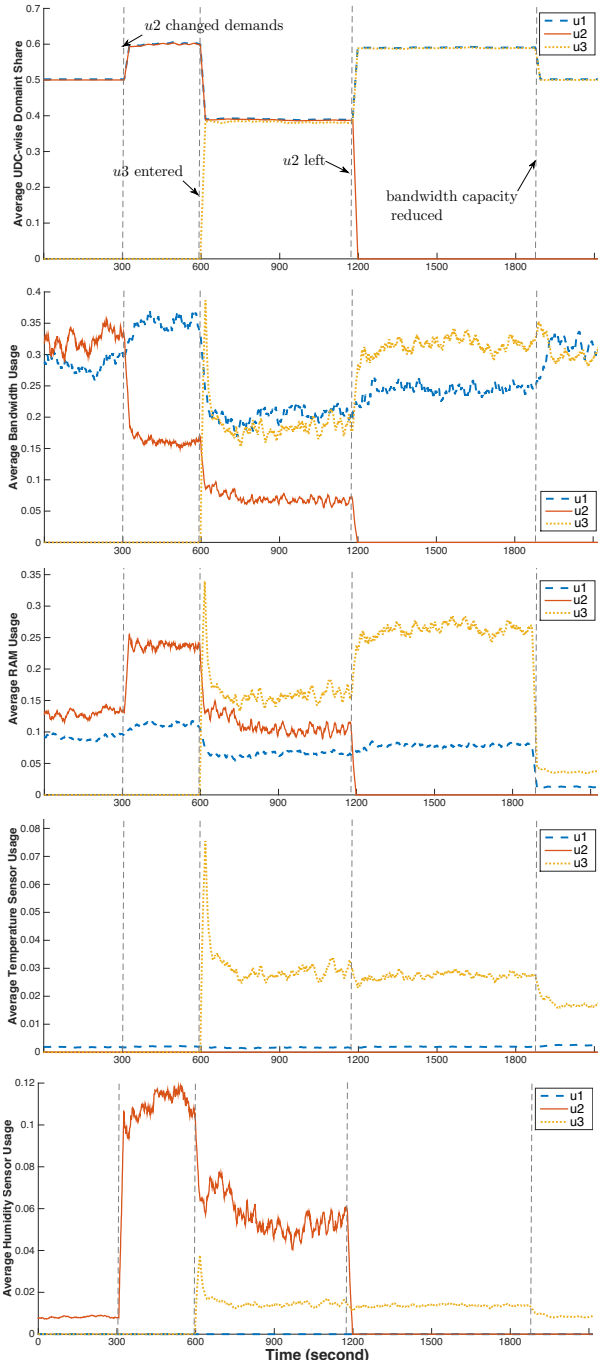

Fig. 5. Symbiot resource allocation in a dynamic system.

network with the resource requirements of $\langle 0, 200, 20, 20\rangle$ for sensing and $\langle 4, 200, 0, 0\rangle$ for data forwarding. Symbiot lowers the resource allocation for both u_1 and u_2 to accommodate the resource demands for the new users. As u_3 enters the network, the nodes in beginning tries to perform as much sensing as possible until bottleneck nodes are found causing traffic congestion. This will result in gradually lowering the sensing rate as shown between slot 600 and 700. At slot 900 when u_2 leaves the network, Symbiot allocates the released resources from u_2 to u_1 and u_3. At time slot 1900, we intentionally reduce the bandwidth capacities of all nodes by 50%. Symbiot automatically adjusts the allocations for both u_1 and u_3.

B. Symbiot vs Alternative Multi-Resource Allocation

To compare the performance of Symbiot with another multi-resource allocation algorithm, we implemented a naive version of distributed DRF. Here, each node runs the *progressive filling* of DRF [4]. However, the sensing and data forwarding tasks are not traffic-aware. If any node receives data packets more than $MaxQ_n^u$, the node will drop the packets. In these experiments, we measured the completion times needed to perform sensing tasks on all the nodes required by the users.

Fig. 6 presents the reductions of average completion times in networks with different numbers of gateways. Here, the network has two users with sensing requirements of $\langle 0, 1, 2, 0\rangle$ and $\langle 0, 10, 0, 20\rangle$ and forwarding requirements of $\langle 1, 20, 0, 0\rangle$ and $\langle 2, 50, 0, 0\rangle$. Fig. 7 shows Symbiot outperforms the implemented distributed DRF in networks with different numbers of users. The used topology had 5 gateways and the sensing and forwarding requirements were randomly generated.

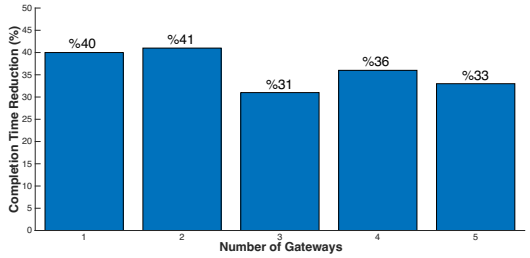

Fig. 6. Average job completion time saving that Symbiot achieved against naive, distributed DRF in networks with different numbers of gateways.

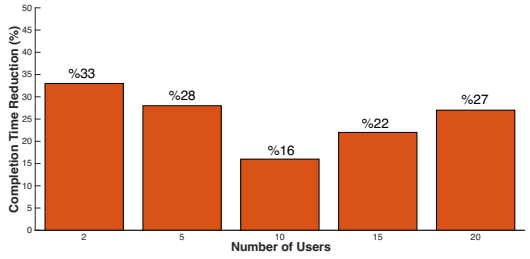

Fig. 7. Average job completion time saving that Symbiot achieved against the implemented version of distributed DRF in networks with different users.

V. Related Work

One way to ensure fairness in multi-resource systems is to employ single-resource abstraction models where system resources are divided into fixed partitions, often referred as *slots*. As pointed out in [4], employing such simple abstraction models are considered to be inefficient when users have heterogeneous demands for resources. Ghodsi *et al*. [4] propose Dominant Resource Fairness (DRF) as an alternative approach to ensure fairness in multi-resource systems. DRF satisfies several highly desirable fairness properties, and it quickly received significant attention from the research community. Parkes *et al*. [9] extend the DRF paradigm to provide a compelling solution for weighted and zero demands users. In fact, the node-wise dominant share computations in SymbIoT is greatly inspired by [9] work. Wang *et al*. [10] suggest to the use of the notion of *global dominant shares* to address the heterogeneity of server capabilities in cloud computing. All the approaches above focus on systems with centralized schedulers. As discussed in Section I, deploying centralized schedulers can be very expensive and not feasible for networks with time-varying resources. We purpose a solution to distributedly allocating resources for users. We also show how indirect resource requirements coming from multi-hop communications should be handled in multi-resource scheduling. This is crucial for ensuring multi-resource fairness across the whole network.

VI. Conclusion

This paper addresses the problem of *multi-resource* fairness between users in data gathering applications in WSNs and IoT. Here, nodes can offer multiple types of resources with heterogeneous, time-varying capacities. Users can have different requirements on various resource types. We explain why the current state-of-the-art falls short when it comes to achieve multi-resource fairness in such networks. By utilizing graph theory, queueing theory and dominant resource sharing, we propose *Symbiot*, a lightweight, distributed algorithm as a solution to this problem. Our simulations based on Cooja network emulator shows the practical performance of Symbiot.

References

[1] Z. Sheng, S. Yang, Y. Yu, A. Vasilakos, J. Mccann, and K. Leung, "A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities," *IEEE Wireless Commun.*, vol. 20, no. 6, pp. 91–98, 2013.

[2] S. Yang, U. Adeel, and J. McCann, "Selfish mules: Social profit maximization in sparse sensornets using rationally-selfish human relays," *IEEE JSAC*, vol. 31, no. 6, pp. 1124–1134, 2013.

[3] S. Yang, X. Yang, J. A. McCann, T. Zhang, G. Liu, and Z. Liu, "Distributed networking in autonomic solar powered wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 750–761, 2013.

[4] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant Resource Fairness: Fair allocation of multiple resource types," in *Proc. USENIX NSDI*, 2011.

[5] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "Faircloud: Sharing the network in cloud computing," in *Proc. ACM SIGCOMM*, 2012.

[6] S. Yang and J. A. McCann, "Distributed optimal lexicographic max-min rate allocation in solar-powered wireless sensor networks," *ACM Trans. Sensor Networks*, vol. 11, no. 1, p. 9, 2014.

[7] "Contiki: The Open Source OS for the Internet of Things." [Online]. Available: http://www.contiki-os.org

[8] A. Sridharan and B. Krishnamachari, "Explicit and precise rate control for wireless sensor networks," in *Proc. ACM SenSys*, 2009, pp. 29–42.

[9] D. Parkes, A. Procaccia, and N. Shah, "Beyond Dominant Resource Fairness: Extensions, limitations, and indivisibilities," in *Proc. ACM EC*, 2012.

[10] W. Wang, B. Li, and B. Liang, "Dominant Resource Fairness in cloud computing systems with heterogeneous servers," in *Proc. IEEE INFOCOM*, 2014.